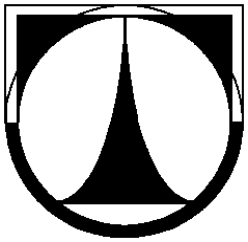
TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta strojní

Katedra aplikované kybernetiky

Automatizované systémy řízení ve strojírenství



Diplomová práce

Systém pro evidenci a vyhodnocení informací

Liberec, 2007 Petr Veselý

Anotace

Systém pro evidenci a vyhodnocení informací

Práce se zabývá návrhem a implementací evidenčního systému pro správu klientů v prostředí realitní kanceláře a následné práci s těmito údaji. Řešení je navrženo pro platformu MS Windows a MS Office. V první části je stručně popsána teorie návrhu databází a základní popis programovacího jazyka Visual Basic for Application. V části experimentální je popsán samotný návrh aplikace a to z pohledu procesního, datového a programového. Dále jsou zde uvedeny vybrané příklady jednotlivých částí programu s jejich stručným popisem.

Abstract

The system for evidence and evaluation of information

The thesis deals with the suggestion and implementation of the evidence system for administration of clients in the environment of the real estate agent and the following operation with these data. The solution is designed for the MS Windows and the MS Office platform. The first part deals with the theory of the database design and the basic description of the programming language Visual Basic for Application. The application project itself is described in the experimental part - from the process, data and programme views. Examples of individual programme parts are also included with their brief description.

Klíčová slova

Databáze, Visual Basic for Application, Access, Excel, Systém, Programování

Keywords

Database, Visual Basic for Application, Access, Excel, System, Programing

Prohlášení

Prohlašuji, že jsem diplomovou práci s názvem: „Systém pro evidenci a vyhodnocení informací" vypracoval samostatně pod vedením Prof. Ing. Miroslava Olehly, CSc., s použitím literatury, uvedené na konci mé diplomové práce v seznamu použité literatury.

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo) a § 35 (o nevýdělečném užití díla k vnitřní potřebě školy).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé práce a prohlašuji, že souhlasím s případným užitím mé práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne vyžadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

V Liberci dne: Petr Veselý

……..

Poděkování

Předně bych chtěl velice poděkovat vedoucímu mé diplomové práce profesoru Miroslavu Olehlovi, svému konzultantovi inženýru Markovi Pašťalkovi a také kolegům, kteří mi pomáhali, když se práce nedařila.

Seznam použitých symbolů

MS	-	Microsoft
SOSB	-	Smlouva o smlouvě budoucí
Zákazník	-	divize Development firmy Metrostav a.s.
VBA	-	Visual Basic for Application

Obsah

1 Úvod

Z potřeb divize Development nevětší české stavební společnosti METROSTAV a.s. vznikl požadavek na vytvoření programu, ve kterém by bylo možno evidovat klienty divize, který by umožnil sledovat jednotlivé developerské projekty, generoval smlouvy SOSB (smlouva o smlouvě budoucí) a vyhodnocoval platební morálku jednotlivých klientů. Mezi další požadavky zadavatele patřila především práce více uživatelů se systémem, komunikace s dalšími programy zadavatele a možnost jeho dalšího rozšiřování.

Před samotným návrhem programu je třeba provést datový, procesní a programový návrh takovéhoto systému. Tento návrh spočívá v definici dat, vazeb a procesů, které by měl systém obsahovat. Proto je nutno nejprve zjistit všechny požadavky potencionálních uživatelů a dle zjištěných informací, přesně definovat prostup informací systémem, jejich logické vazby závislé na typu, charakteru a jejich časové souslednosti. Dalším krokem je výběr vhodného vývojového prostředí pro tvorbu aplikace a výběr datového úložiště. Dále je třeba vytvořit vlastní strukturu programu s ohledem na jeho funkční nezávislost. Závěrečným krokem je pak otestování aplikace na reálných datech, jeho ladění a případná oprava chyb. Poté je program předán zadavateli a po ověření požadované funkčnosti z jeho strany je program uvolněn do běžného (tzv. „ostrého“) provozu.

2 Teoretická část

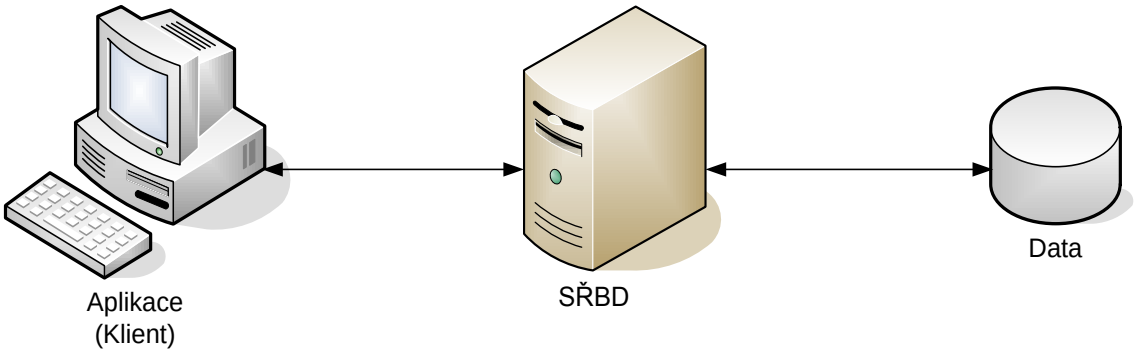
2.1 Úvod do problematiky zpracování dat

S rozvojem lidského poznání roste prudce množství informací, které tento proces vyžaduje a také produkuje. Pro efektivní práci s informacemi začaly vznikat specializované informační systémy. Můžeme je definovat např. jako „systémy pro sběr, uchovávání, vyhledávání a zpracovávání informací (údajů, dat) za účelem jejich poskytování." Tvorbou informačních systémů se zabývá vědní obor informatika, vydělený z oboru kybernetika. [4]

Rozvoj informačních systémů je úzce spjat s rozvojem výpočetní techniky, zejména počítačů. Od svých počátků byla využívána na zpracování velkých informačních objemů na jednom počítači. Takové systémy obvykle nazýváme systémy hromadného zpracování dat nebo agendové zpracování. Data se nejprve ručně zaznamenávala na stanovené formuláře, dále se přepisovala na vhodné médium (děrné štítky, diskety), následovalo primární a sekundární zpracování, výsledkem byly vytištěné výstupní sestavy. Celková doba tohoto zpracování byla poměrně dlouhá, proto nebylo možno tímto způsobem vyhodnocovat dynamické děje (typickou aplikací je měsíčně zpracovávaná mzdová agenda). Pro zpracovávání se používaly účelově vyvinuté programy, většinou v jazyce Cobol. Příznačná byla úzká provázanost fyzických a logických datových struktur a odtud plynoucí nesnadnost získat údaje v jiné podobě, než pro jakou byla úloha vytvořena.

Snahy odstranit nevýhody agendového zpracování vedly k oddělení dat od programu. Data jsou uložena samostatně v bázi dat a programy si vybírají potřebné informace. Na tomto principu pracují databázové systémy. Počátky databázových systémů spadají do 60.

let, proti agendovému zpracovávání dat představují nový kvalitativní stupeň. Databázový systém vzniká spojením systému řízení báze dat (SŘBD) a vlastní báze dat.



Obr. 1 Databázový systém

2.2 Základní pojmy databázového systému

Systém řízení báze dat (SŘBD) lze chápat jako souhrn procedur a datových struktur, které zajišťují nezávislost databázových aplikací na detailech vytváření, výběru, uchování, modifikaci a zabezpečení ochrany databází na fyzických paměťových strukturách počítače. Pro práci s daty SŘBD podporuje zejména tyto funkce:

- vytvoření báze dat (CREATE),
- vkládání dat (INSERT),
- aktualizace dat (UPDATE),
- rušení dat (DELETE),
- výběr z báze dat (SELECT).

Dále jsou zde uvedeny některé základní pojmy, se kterými se budeme setkávat v následujícím textu.

Data jsou údaje, které mají určitou vypovídající schopnost. Mohou být určitým způsobem uspořádány (seřazeny např. podle velikosti, chronologicky atd.) a jsou uživateli k dispozici v různých formách (tabulky, grafy, zvukové signály, grafická forma atd.). Data jsou obvykle rozdělena

na dílčí údaje (atributy) o dané množině objektů (entit), na základě nichž lze získat určitou informaci, která může vést k rozhodovacímu procesu.

Záznam je souhrn údajů (atributů) o dané části objektu, které jsou uloženy v položkách (polích, Fields) charakterizovaných názvem a datovým typem. Význam pojmu záznam (Record) a položka lze snadno ukázat na bázi dat KLIENT. Abychom mohli u každého klienta zaznamenat potřebné údaje, musí mít záznam několik položek, které mohou mít tyto názvy :

- identifikační číslo
- jméno
- příjmení
- datum narození
- bydliště
- telefon

Pozn.: v relačním (tabulkovém) modelu jsou záznamy tvořeny jednotlivými řádky tabulky, položky pak jednotlivými sloupci tabulky.

Datové typy Každá položka musí být určitého datového typu. Obecně akceptované jsou tyto typy dat:

Textový typ	textový řetězec, zpravidla do max. délky 255 znaků.
Číselné typy	pro uložení celých a reálných čísel s pevnou i plovoucí desetinnou tečkou.
Logický typ	k uložení logické hodnoty Ano/Ne (True/False, Yes/No).
Memo	pro uložení textu proměnné délky.
Datumový typ	pro uložení datumu nebo datumu a času.

Nejsou zde uvedeny konkrétní názvy jednotlivých datových typů, protože v jednotlivých SŘBD se tyto názvy, jakož i některé vlastnosti těchto typů odlišují (např. číselné rozsahy, max. délka řetězce atd.).

2.3 Datové modely

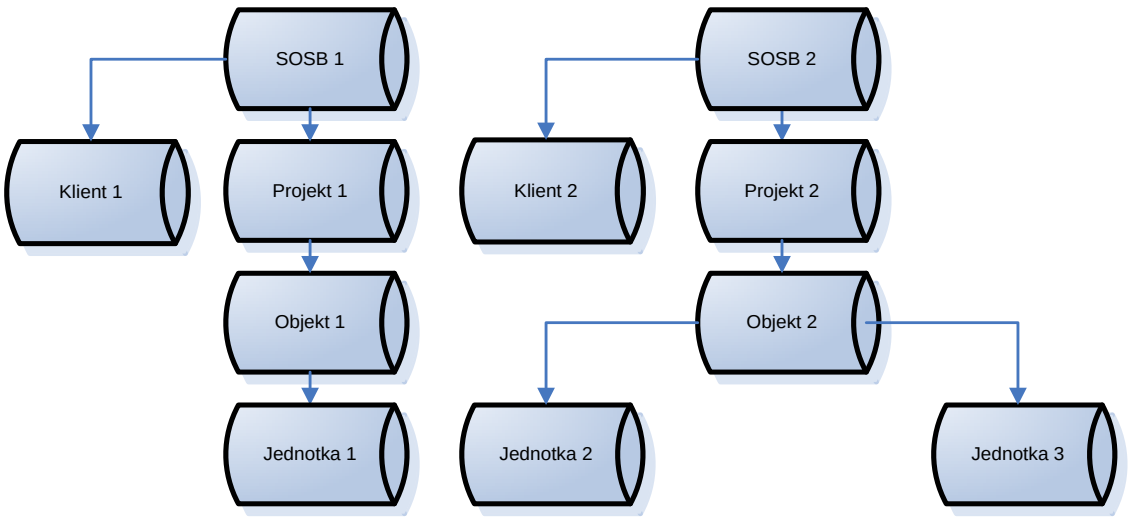
Model je souhrn pravidel pro reprezentaci logické organizace dat v databázi. Rozeznáváme tři základní modely dat:

- Hierarchický
- Síťový
- Relační

Nejnovější a zároveň nejpoužívanější je relační model, který odstraňuje některé nedostatky ostatních modelů.

2.3.1 Hierarchický model dat

Data jsou organizována do stromové struktury. Každý záznam představuje uzel ve stromové struktuře. Vzájemný vztah mezi záznamy je typu rodič/potomek.



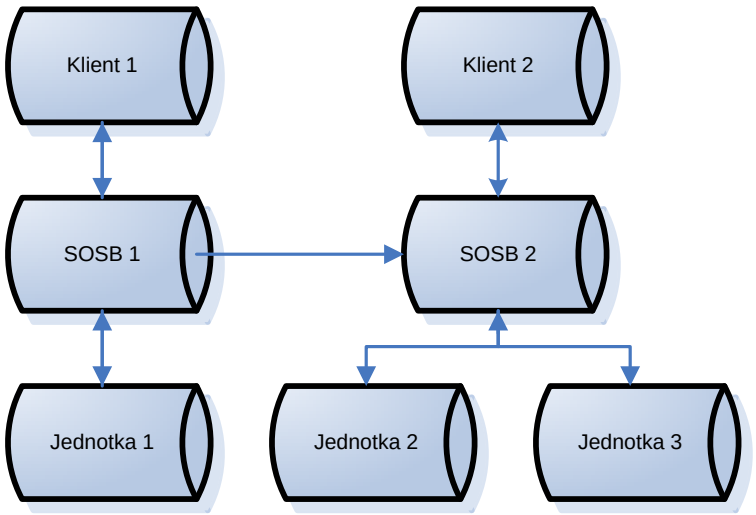
Obr. 2 Hierarchický model dat

Použití hierarchického modelu je vhodné tam, kde i zájmová realita má hierarchickou strukturu. Nalezení dat v hierarchické databázi vyžaduje navigaci přes záznamy směrem dolů (potomek), nahoru (rodič) a do strany (další potomek). Mezi nevýhody hierachického modelu patří:

- v některých případech nepřirozená organizace dat (zejména obtížné znázornění vztahu M:N, který se řeší např. pomocí virtuálních záznamů)
- složité operace vkládání a rušení záznamů.

2.3.2 Síťový model dat

Síťový model dat je v podstatě zobecněním hierarchického modelu dat, který doplňuje o mnohonásobné vztahy. Tyto vztahy jsou označovány jako C-množiny neboli Sets (dále budeme používat pojem set, pro který neexistuje ekvivalentní český výraz). Tyto sety propojují záznamy různého či stejného typu, přičemž spojení může být realizováno na jeden nebo více záznamů.



Obr. 3 Síťový model dat.

Přístup k propojeným záznamům je přímý bez dalšího vyhledávání, k dispozici jsou tyto operace:

- nalezení záznamu podle klíče
- posun na prvního potomka v dílčím setu
- posun stranou na dalšího potomka v setu
- posun nahoru z potomka na jeho rodiče

Nevýhodou síťové databáze je zejména nepružnost a obtížná změna její struktury.

2.3.3 Relační model dat

Nejmladším databázovým modelem je model relační, který byl popsán v roce 1970 Dr. Coddem. V současnosti je tento model nejčastěji využíván u komerčních SŘBD. Relační databázový model má jednoduchou strukturu. Data jsou organizována v tabulkách, které se skládají z řádků a sloupců. Všechny databázové operace jsou prováděny na těchto tabulkách. Dr. Codd definoval jako minimalisticky relační ty systémy, které splňují tyto dvě vlastnosti:

- Databáze je chápana uživatelem jako množina relací a nic jiného.
- V relačním SŘBD jsou k dispozici minimálně operace selekce, projekce a spojení, aniž by se vyžadovaly explicitně předdefinované přístupové cesty pro realizaci těchto operací.

Dále definoval Dr. Codd dalších 12 pravidel pro relační SŘBD:

1. **Informační pravidlo**

   Všechny informace v relační databázi jsou vyjádřeny explicitně na logické úrovni jediným způsobem - hodnotami v tabulkách.

2. **Pravidlo jistoty**

   Všechna data v relační databázi jsou zaručeně přístupná kombinací jména tabulky s hodnotami primárního klíče a jménem

sloupce.

3. **Systematické zpracování nulových hodnot**

   Nulové hodnoty jsou plně podporovány relačním SŘBD pro reprezentaci informace, která není definována, a to nezávisle na datovém typu.

4. **Dynamický on-line katalog založený na relačním modelu**

   Popis databáze je vyjádřen na logické úrovni stejným způsobem jako zákaznická data, takže autorizovaný uživatel může aplikovat stejný relační jazyk ke svému dotazu jako uživatel při práci s daty.

5. **Obsáhlý datový podjazyk**

   Relační systém může podporovat několik jazyků a různých módů použitých při provozu terminálu. Nicméně musí být nejméně jeden příkazový jazyk s dobře definovanou syntaxí, který obsáhle podporuje definici dat, definici pohledů, manipulaci s daty jak interaktivně, tak programem, integritní omezení, autorizovaný přístup k databázi, transakční příkazy apod.

6. **Pravidlo vytvoření pohledů**

   Všechny pohledy, které jsou teoreticky možné, jsou také systémem vytvořitelné.

7. **Schopnost vkládání, vytvoření a mazání**

   Schopnost zachování relačních pravidel u základních i odvozených relací je zachována nejen při pohledu na data, ale i při operacích průniku, přidání a mazání dat.

8. **Fyzická datová nezávislost**

   Aplikační programy jsou nezávislé na fyzické datové struktuře.

9. **Logická datová nezávislost**

   Aplikační programy jsou nezávislé na změnách v logické struktuře databázového souboru.

10. Integritní nezávislost

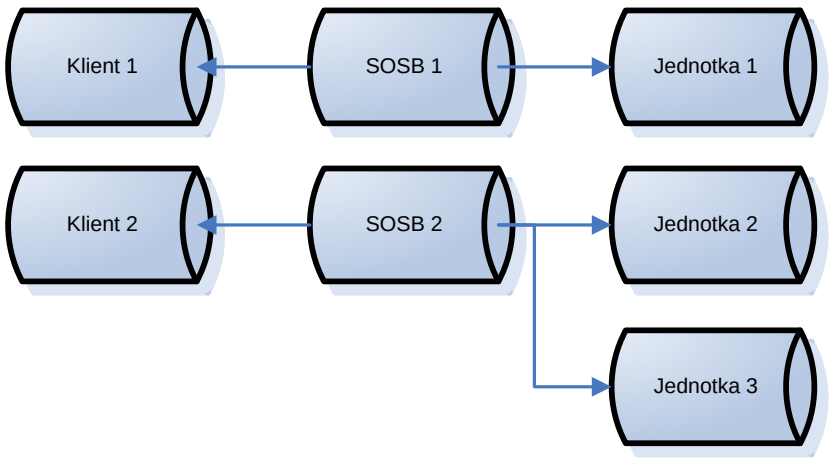
Integritní omezení se musí dát definovat prostředky relační databáze nebo jejím jazykem a musí být schopna uložení v katalogu, a nikoliv v aplikačním programu.

11. Nezávislost disribuce

Relační SŘBD musí být schopny implementace na jiných počítačových architekturách.

12. Pravidlo přístupu do databáze

Jestliže má relační systém jazyk nízké úrovně, pak tato úroveň nemůže být použita k vytváření integritních omezení a je nutno vyjádřit se v relačním jazyce vyšší úrovně. [4]

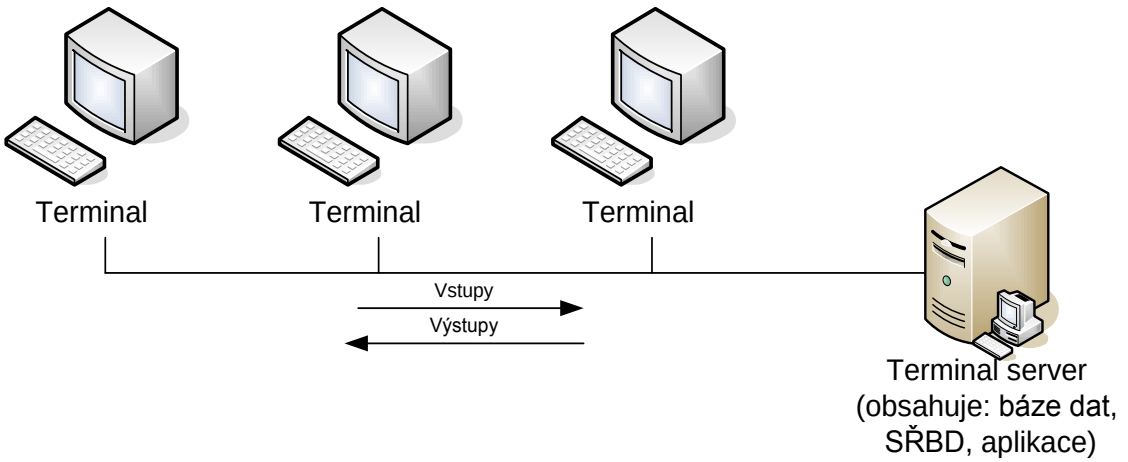


Obr. 4 Relační model dat

2.4 Architektura databází

2.4.1 Centrální architektura

V této architektuře jsou data i SŘBD v centrálním počítači. Tato architektura je typická pro terminálovou síť, kdy se po síti přenášejí vstupní údaje z terminálu na centrální počítač do příslušné aplikace. Výstupy z této aplikace se přenášejí na terminál. Protože aplikační program i vlastní zpracování probíhá na centrálním počítači, který může zpracovávat více úloh, mají odezvy na dotazy určité zpoždění.



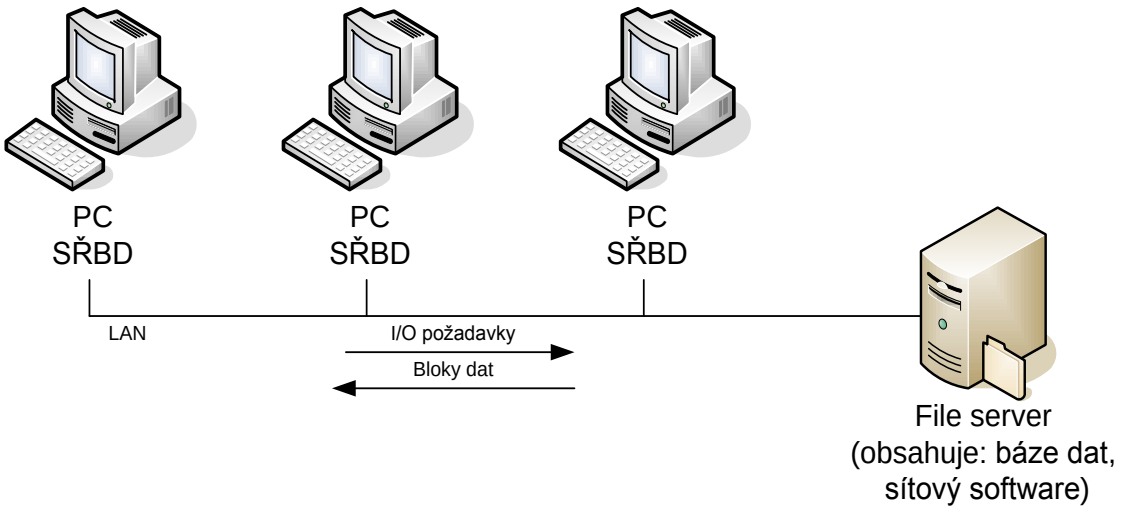
Obr. 5 Centrální architektura databáze

2.4.2 Architektura file-server

Tato metoda souvisí zejména s rozšířením osobních počítačů a sítí LAN. SŘBD a příslušné databázové aplikace jsou provozovány na jednotlivých počítačích, data jsou umístěna na file-serveru a mohou být sdílena. Aby nedocházelo ke kolizím při přístupu více uživatelů k jedněm datům, musí SŘBD používat vhodný systém zamykání

(položek nebo celých tabulek). Komunikace uživatele se systémem probíhá následujícím způsobem:

1. uživatel zadá dotaz
2. SŘBD přijme dotaz, zasílá požadavky na data file-serveru
3. file-server posílá bloky dat na lokální počítač, kde jsou data
4. zpracovávána podle zadaného dotazu (vyhledávání, setřídění atd.)
5. výsledek dotazu se zobrazí se na obrazovce osobního počítače



Obr. 6 Archiktetura file-server

2.4.3 Architektura klient-server

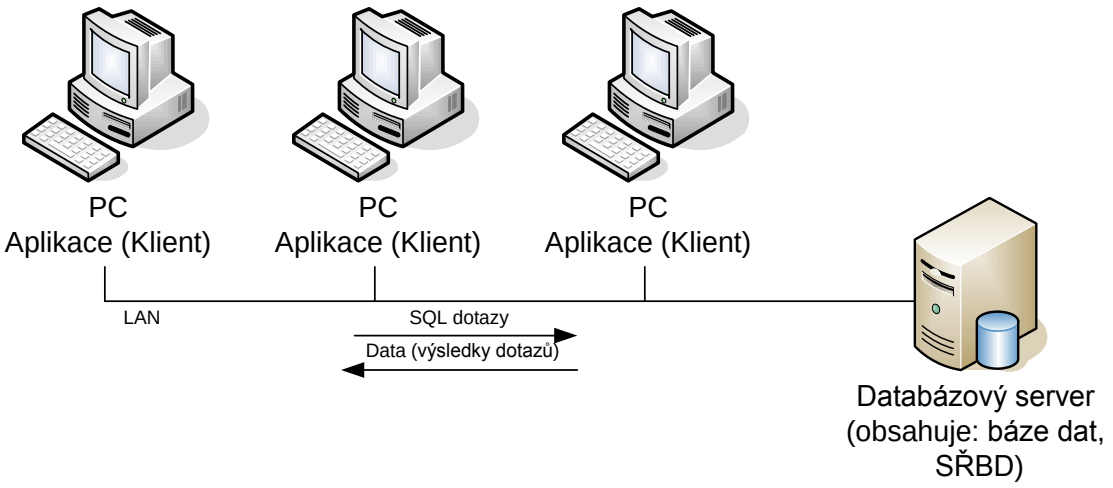
V podstatě je založena na lokální síti (LAN), personálních počítačích a databázovém serveru. Na personálních počítačích běží program podporující např. vstup dat, formulaci dotazu atd. Dotaz se dále předává pomocí jazyka SQL (Structured Query Language) na databázový server, který jej vykoná a vrátí výsledky zpět na personální počítač. Databázový server je tedy nejvíce zatíženým prvkem systému a musí být tvořen dostatečně výkonným počítačem. Celá komunikace probíhá tímto způsobem:

1. uživatel zadává dotaz (buď přímo v SQL, nebo musí být do tohoto jazyka přeložen)

2. dotaz je odeslán na databázový server
3. databázový server vykoná dotaz
4. výsledek dotazu je poslán zpět na vysílací počítač, kde je zobrazen

Architektura klient-server redukuje přenos dat po síti, protože dotazy jsou prováděny přímo na databázovém serveru a na personální počítač jsou posílány pouze výsledky. Např. pokud je mezi 10 000 záznamy pouze 100 záznamů, které splňují podmínku dotazu, pak na personální počítač putuje pouze těchto 100 záznamů. V případě architektury file-server je však nutné poslat všech 10 000 záznamů na personální počítač, tam se teprve provede dotaz a zpracuje nalezených 100 záznamů.

Architektura klient-server vyhovuje i náročným aplikacím a je využívána většinou renomovaných databázových firem.



Obr. 7 Archiktetura klient-server

Kromě jazyka SQL, který představuje standardní dotazovací jazyk, existují ještě další standardy pro navazování komunikace mezi aplikacemi ještě před vlastním zahájením komunikace v SQL.

2.4.4 Distribuovaná databáze

Distribuovaná databáze je množina databází, která je uložena na několika počítačích. Uživateli se však jeví jako jedna velká databáze. Distribuovanou databázi charakterizujeme třemi vlastnostmi:

Transparentnost z pohledu klienta se zdá, že všechna data jsou zpracovávána na jednom serveru v lokální databázi. Uživatel používá syntakticky shodné příkazy pro lokální i vzdálená data, nespecifikuje místo uložení dat, o to se stará distribuovaný SŘBD.

Autonomnost s každou lokální bází dat zapojenou do distribuované databáze je možno pracovat nezávisle na ostatních databázích. Lokální databáze je funkčně samostatná, propojení do jiné části distribuované databáze se v případě potřeby zřizují dynamicky. V distribuované databázi neexistuje žádný centrální uzel nebo proces odpovědný za vrcholové řízení funkcí celého systému, což výrazně zvyšuje odolnost systému proti výpadkům jeho částí.

Nezávislost na počítačové síti jsou podporovány různé typy architektur lokálních i globálních počítačových sítí (LAN, WAN). V jedné distribuované databázi tedy mohou být zapojeny počítače i počítačové sítě různých architektur, pro komunikaci se používá jazyk SQL. [4]

2.5 Datová analýza

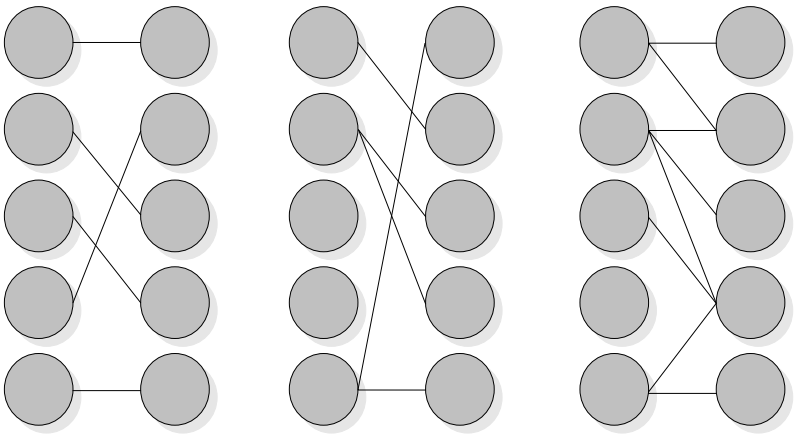
Nejdůležitější částí návrhu databázového systému je volba vhodného uložení dat, definice tabulek a jejich položek, včetně definice vazebních vztahů mezi nimi. Proces jejich návrhu označujeme pojmem datová analýza.

2.5.1 Entitní vztahy

Entita je objekt, o kterém chceme v bázi dat registrovat nějaké údaje (např. JEDNOTKA, KLIENT, SOSB apod.). Pro návrh logické struktury báze dat jsou důležité vzájemné vztahy mezi různými typy entit - tzv. entitní vztahy. Prakticky zkoumáme pouze vztahy mezi dvojicí typů entit - tzv. binární entitní vztahy. Existují 3 typy těchto vztahů.

Máme-li dva typy entit X, Y a množiny jejich výskytu F(X) a F(Y). Pak nastávají tři možné stavy:

1. Mezi typy entit X, Y je vztah 1:1, pokud existuje funkční zobrazení z F(X) do F(Y) i naopak.
2. Mezi typy entit X, Y je vztah 1:N, pokud existuje funkční zobrazení z F(Y) do F(X).
3. Mezi typy entit X, Y je vztah M:N, pokud neexistuje funkční zobrazení z F(X) do F(Y) a naopak.



Vztah 1:1 Vztah 1:N Vztah M:N

Obr. 8 Vztahy mezi entitami

Vztah 1:N existuje např. mezi typy entit KLIENT-SOSB – kdy jeden klient může mít více smluv SOSB. Vztah 1:1 nastává např. mezi typy entit JEDNOTKA-SOSB, protože jednotka může být prodána pouze v jedné SOSB. Vztah M:N nastává např. mezi typy entit KLIENT-REZERVACE, kdy rezervace na jednu jednotku může být provedena více klienty a také jeden klient může rezervovat více jednotek.

2.5.2 Úrovně abstraktního pohledu na data

První SŘBD, které vznikaly na konci 60. let, se vyznačovaly úzkou provázaností fyzického a logického formátu dat. U novějších SŘBD pak dochází k hierarchickému rozvrstvení dat do těchto úrovní, přičemž jednotlivé úrovně jsou relativně nezávislé. Nejdůležitější je zejména nezávislost logického schématu báze dat od interního a fyzického schématu. [4]

Fyzické schéma úzce souvisí s použitým operačním systémem (konkrétní organizace souborů na disku, jejich rozložení na sektory a clustry určité délky atd.).

Interní data jsou uložena v typových souborech, přístup k

schéma jednotlivým větám souborů je organizován vhodným mechanismem (primární a sekundární indexy, Bayerovy stromy atd.).

Logické schéma vzniká implementací konceptuálního modelu do konkrétního SŘBD (návrh struktury datových vět). Struktura tohoto schématu je určena použitým datovým modelem v daném SŘBD (hierarchický, síťový, relační).

Externí schéma je rozdílné pro každou skupinu uživatelů. Umožňuje virtuální pohledy na zvolenou část báze dat (pomocí konkrétních formulářů, výstupních sestav, ale také přístupových práv k datům).

2.6 Relační databáze

Na osobních počítačích se dnes provozují prakticky výhradně SŘBD s relační architekturou, proto jí budeme věnovat větší pozornost.

Relací druhého stupně rozumíme kartézský součin dvou neprázdných množin. Znázorníme jej formou tabulky, která má m řádků a n sloupců. V komerčně používaných SŘBD se místo pojmu relace používa pojem tabulka (Table), aby došlo k odlišení od matematického aparátu.

Je nezbytné, aby každá tabulka v databázi měla své jedinečné jméno. Záznam (Record) jako souhrn údajů o daném objektu je v tabulce reprezentován jedním řádkem tabulky. Sloupec tabulky definuje jednu položku (Field). Jak již bylo řečeno, každé pole musí mít svůj název a musí být určitého datového typu.

Primární klíč je taková podmnožina položek, která má nezávisle na čase tu vlastnost, že jednoznačně identifikuje každý záznam relace. Z toho je zřejmé, že primární klíč relace je neredundandní. V tabulce vždy existuje alespoň jeden primární klíč, který je v

nejhorším případě tvořen všemi položkami dané tabulky. Řada SŘBD umožňuje vytvořit zvláštní položku, která nabývá hodnot pořadových čísel záznamů, v některých případech je tato položka vhodná jako primární klíč.

Z popisu tabulkového vyjádření relace vyplývají tyto vlastnosti:

1. Homogenita sloupců - v každém sloupci jsou všechny položky stejného typu.
2. V relaci neexistují dva stejné řádky.
3. Pořadí řádků je nevýznamné, protože jednotlivé řádky jsou identifikovatelné pomocí primárního klíče.
4. Pořadí sloupců (položek) je nevýznamné, protože sloupce jsou označeny názvem položky.

2.7 Visual Basic for Application

2.7.1 Základní popis jazyka

Každý program používaný na počítači musel někdo naprogramovat. Někdy jsou to až milióny řádků přesných instrukcí, podle kterých se počítač řídí. K tomuto účelu slouží programovací jazyk.

Programovací jazyky lze rozdělit na jazyky:

nižší úrovně - umožňují tvorbu stabilnějších a rychlejších aplikací, ovšem za cenu dlouhých programových kódů. Mezi tyto jazyky patří Assembler nebo C++.

vyšší úrovně - výhodou je mnohem kratší doba přípravy aplikace, nevýhodou pak pomalejší chod programu. Kromě Visual Basicu do této skupiny patří například Power Builder.

Visual Basic a jeho odnože se v posledních letech staly hlavním programovacím prostředkem na platformě produktů firmy Microsoft. Různé mutace tohoto programovacího jazyka se používají nejen pro programování samostatných (i síťových) aplikací, ale také pro tvorbu maker v balíku programů Microsoft Office (Visual Basic for Applications) a při programování internetových aplikací, spouštěných na straně serveru (ASP) i klienta (Visual Basic Script). V obou případech se zdrojový kód kompiluje až při běhu programu.

K oblibě tohoto programovacího jazyka přispěla hlavně jednoduchá syntaxe, snadná tvorba uživatelského rozhraní a velká variabilita a flexibilita jazyka. Programování ve Visual Basicu je počítáno mezi objektově orientované a událostmi řízené techniky. Stručně řečeno to znamená: Programátor může používat velké množství předdefinovaných objektů, jako jsou formuláře, textová

pole pro zadávání a zobrazování dat, příkazová tlačítka, menu, popisky a velké množství dalších objektů. Souhrnně tyto objekty nazýváme ovládacími prvky (anglicky controls).

Každý ovládací prvek má pak definovány své vlastnosti, metody a události.

Vlastnosti prvku jsou přesně v souladu s názvem vlastnosti daného ovládacího prvku. Tyto vlastnosti udávají vzhled a chování ovládacího prvku v aplikaci. U textového pole lze například definovat font, kterým bude v poli zobrazen text, velikost pole a jeho umístění na formuláři, název, pomocí něhož se na ovládací prvek bude odkazovat programový kód a mnoho dalších vlastností.

Metody ovládacích prvků vlastně představují činnosti, které daný ovládací prvek může vykonávat, případně které mohou být vykonány na něm. Pokud se podržíme příkladu textového pole, najdeme u něj definované např. metody Refresh (obnovit zobrazovaná data) nebo SetFocus (umístění fokusu – tj. zaměření – na textové pole).

Události definované u všech ovládacích prvků zajišťují, že programování ve Visual Basicu se počítá mezi událostmi řízené programovací techniky. U každého ovládacího prvku najdeme seznam událostí, které mohou při běhu naprogramované aplikace vzniknout v přímém vztahu k tomuto prvku. Klasickými událostmi, jež najdeme u většiny ovládacích prvků, jsou Click (klepnutí myší na ovládací prvek), DblClick (poklepání myší na ovládací prvek), GotFocus a LostFocus (události nabytí, resp. pozbytí fokusu, tj. zaměření) a mnoho dalších. Události slouží k programování procedur, které se vykonají jako odezva na výskyt určité události. Např. procedura, která bude v programovém kódu přiřazena k události Clik na příkazové tlačítko, se vykoná vždy, když uživatel na toto tlačítko klikne myší. [5]

2.7.2 Seznámení s VBA

2.7.2.1 Uživatelské prostředí

První dvě části vývojového prostředí Visual Basic for Application jsou standardní snad ve všech aplikacích Windows. Nabídka (menu) a nástrojová lišta jsou základními ovládacími prvky každého programu. Další podokna vývojového prostředí jsou již plně přizpůsobena potřebám programování. Bílá střední část vývojového prostředí, která zabírá nejvíce místa, je pracovní plocha. Tento sektor slouží ke dvěma základním účelům:

- pro grafické navrhování formulářů a dialogových oken
- pro psaní, editaci a ladění programového kódu

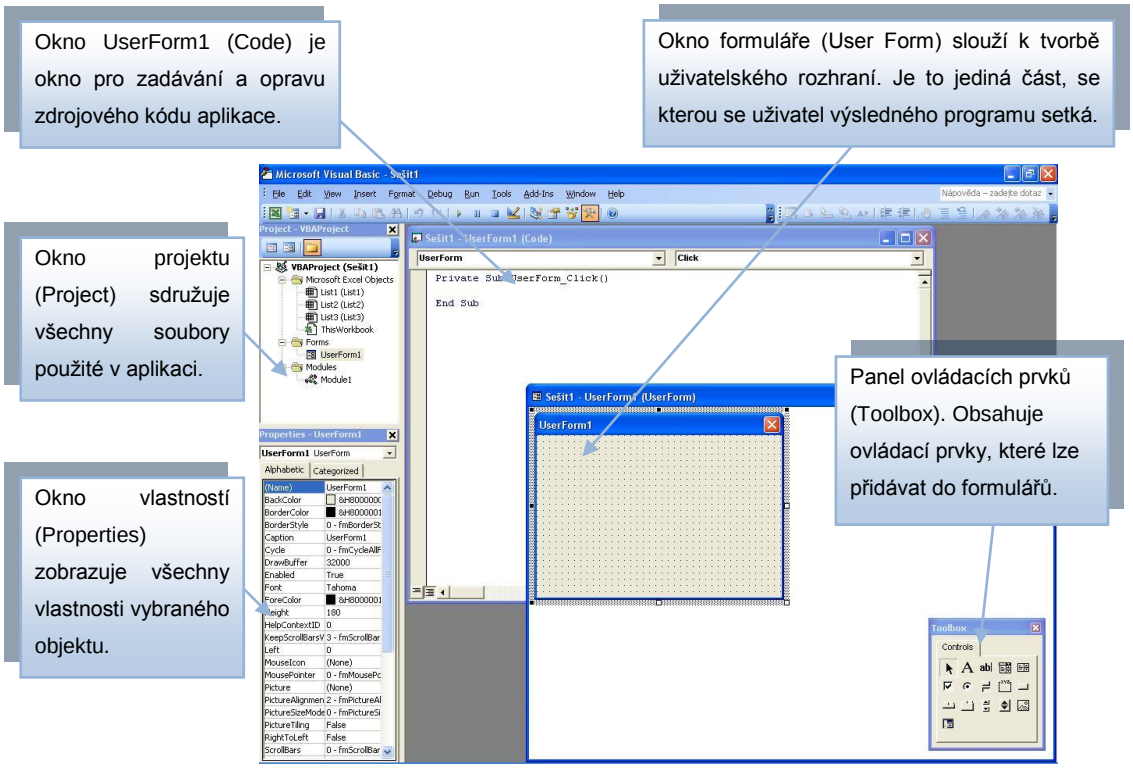
V pravé spodní části je panel ovládacích prvků (ToolBox). Tento panel obsahuje schématické ikony ovládacích prvků, které lze přidávat do formulářů. První ikonou je šipka, jež nepředstavuje ovládací prvek, ale volbu kurzoru pro manipulaci s již vytvořenými objekty.

Dále následují nejpoužívanější ovládací prvky v tomto pořadí: PictureBox (obrázkové pole), Label (popisek), TextBox (textové pole), Frame (rámec), CommandButton (příkazové tlačítko) atd.

V levé horní části je okno projektu. Toto okno obsahuje vždy všechny základní součásti, ze kterých se editovaný projekt skládá. Pokud mluvíme o součástech projektu, nemáme na mysli jednotlivé ovládací prvky vkládané do formulářů. Prvky, které se na formuláře umísťují z panelu ovládacích prvků, tedy v okně projektu, neuvidíme. Okno projektu zobrazuje objekty, které jsou součástí projektu ve dvou úrovních jako stromovou strukturu. Vyšší úrovní je kolekce konkrétních objektů, na niž jsou navázány jednotlivé objekty kolekce.

Následujícím oknem vývojového prostředí umístěným vlevo dole je okno vlastností ovládacího prvku. Aktuálně je v tomto okně

zobrazen soubor vlastností právě zvoleného ovládacího prvku. Vzhledem k tomu, že jediným ovládacím prvkem nově založeného projektu je automaticky založený formulář, obsahuje okno vlastností informace právě o tomto okně. Rozbalovací lišta (ComboBox) v horní části okna slouží ke zvolení ovládacího prvku, jehož soubor vlastností chceme zobrazit. Tam je vždy zobrazen název ovládacího prvku (aktuálně Form1), následovaný udáním typu tohoto prvku v angličtině (aktuálně Form).



Obr. 9 Hlavní okno vývojového prostředí VBA

2.7.2.2 Základní charakteristiky jazyka VBA

Proměnné, datové typy

Proměnná je určité místo v paměti, na které se odkazuje její název. Název proměnné nesmí být delší než 255 znaků, nesmí obsahovat tečky, mezery a musí začínat písmenem. Obsah proměnné se může za běhu programu měnit. Příklad explicitní deklarace proměnné Jmeno:

Dim Jmeno As String

Příkazem Dim oznamujeme programu vytvoření nové proměnné. Příkaz se také postará o přidělení potřebné paměti. Klíčové slovo As String (jako řetězec) je datovým typem pro textový řetězec.

Příklady:

Dim Veta As String (textový řetězec)

Dim Tmp As Boolean (logické hodnoty True nebo False)

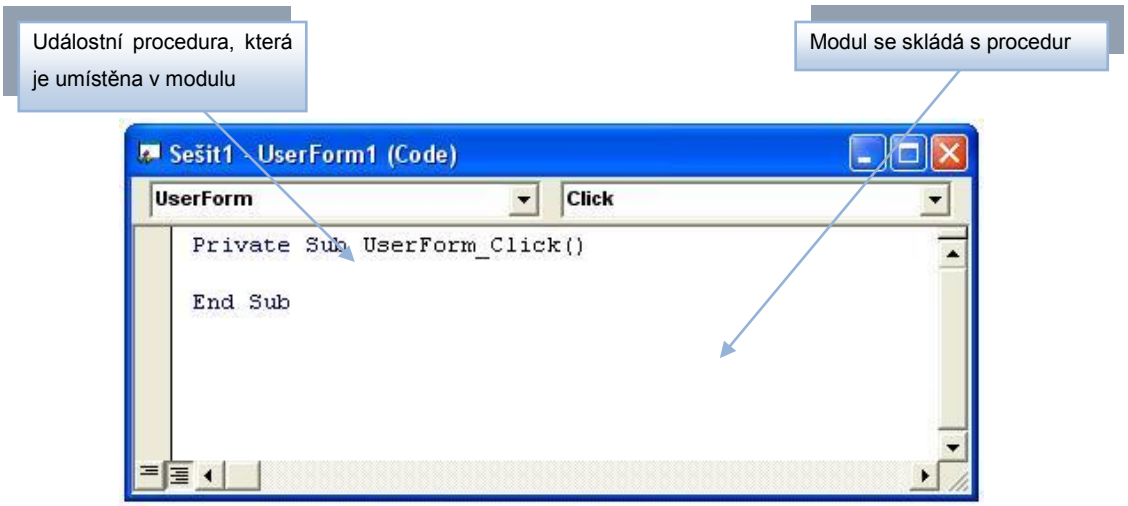
Dim Dnes As Date (datum a čas)

Dim Pole(1 to 5) (pole)

Není-li proměnná jasně (explicitně) deklarována jako jiný datový typ, stane se automaticky typem Variant. Variant je speciální datový typ, který může obsahovat data jakéhokoliv typu. Proměnné, které nejsou deklarovány, program Visual Basic automaticky vytvoří. Výjimku tvoří pole, jež musí být deklarováno vždy. Umístěním příkazu Option Explicit do deklarační části každého nového modulu si explicitní deklaraci proměnných vynutíte.

Platnost proměnných

Deklarací proměnných se také určuje, zda budete s touto proměnou pracovat pouze na úrovni procedury, nebo na úrovni modulu. V modulu jsou definovány všechny procedury.



Obr. 10 Definice procedur

Procedury: *Dim Cislo As Integer*

Proměnná deklarovaná příkazem Dim bude existovat pouze po dobu zpracovávání procedury. Příkaz Dim je možné použít i na úrovni modulu.

Static Cislo As Integer

Příkaz Static uchová hodnoty proměnných po celou dobu běhu programu.

Moduly: *Private Cislo As Integer*

Proměnná deklarovaná příkazem Private je dostupná pouze modulu, v němž je deklarována.

Public Cislo As Integer

Příkazem Public zajistíme použití proměnných ve všech modulech a ve všech procedurách.

Konstanty

Konstanty mají na rozdíl od proměnných neměnnou hodnotu za běhu programu a v paměti nealokují žádný prostor. Lze je používat namísto skutečných hodnot. Konstanty jsou definovány příkazem Const. Je nepsaným pravidlem zapisovat konstanty velkými písmeny. Podobně jako proměnné lze ovlivnit životnost konstant příkazy Public a Private.

Konstanty se rozdělují na uživatelské:

Const MUJ_POZDRAV = "Dobrý, den"

Public Const MOJE_KONSTANTA = 1

Private Const HMOTNOST = 72

a definované systémem

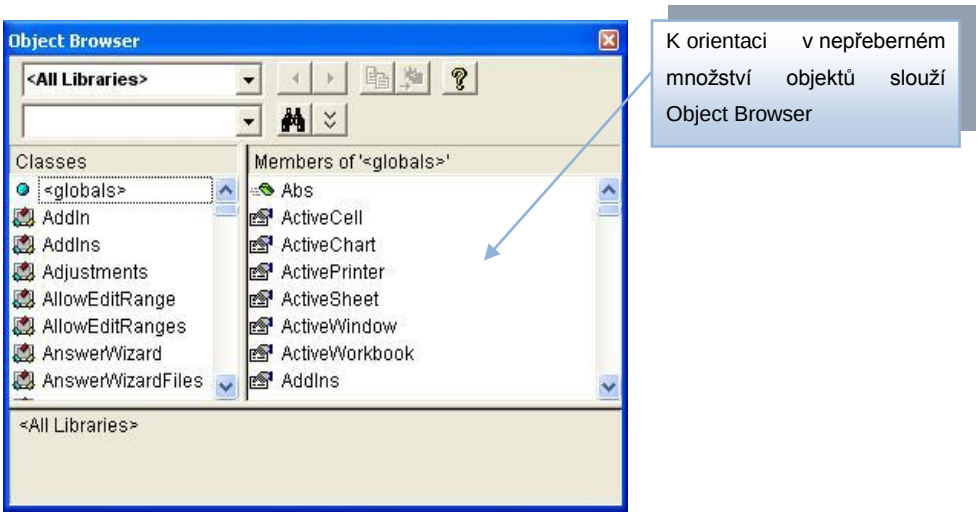
Form1.BackColor = vbBlue

vbBlue je systémem definovaná konstanta modré barvy. Výraz pak změní barvu pozadí formuláře na modrou.

Objekty

Ve Visual Basicu se s objekty setkáme téměř všude. Objekty obsahují kód, který již není třeba psát. Programátor nemusí složitě

programovat tlačítko, jímž by ovládal svoji aplikaci, ale použije jako ovládací prvek tlačítko dodávané s Visual Basic. Ve Visual Basic není objektem pouze tlačítko nebo formulář, ale i tiskárny nebo obrazovka.

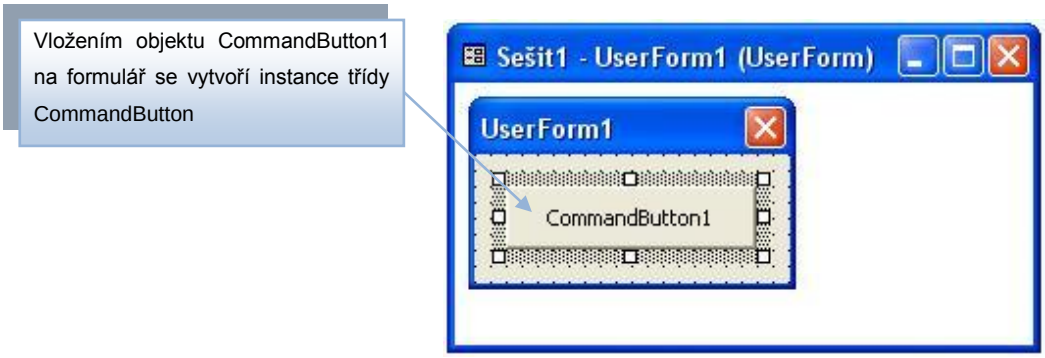


Obr. 11 Práce s objekty v Object Browser

Každý objekt má svoji třídu, svoje vlastnosti, metody a události. Tyto pojmy si v zápětí pokusíme vysvětlit na příkladu CommandButton.

Třídy objektů

Vztah mezi třídou objektu a objektem se dá přirovnat ke vztahu formy na bábovky a bábovkou. Forma je třída, která určuje vlastnosti bábovky, jako je například velikost a tvar. Takovouto formou je po přidání objektu tlačítka na formulář třída CommandButton a jejím výsledkem je instance této třídy s názvem Command1.

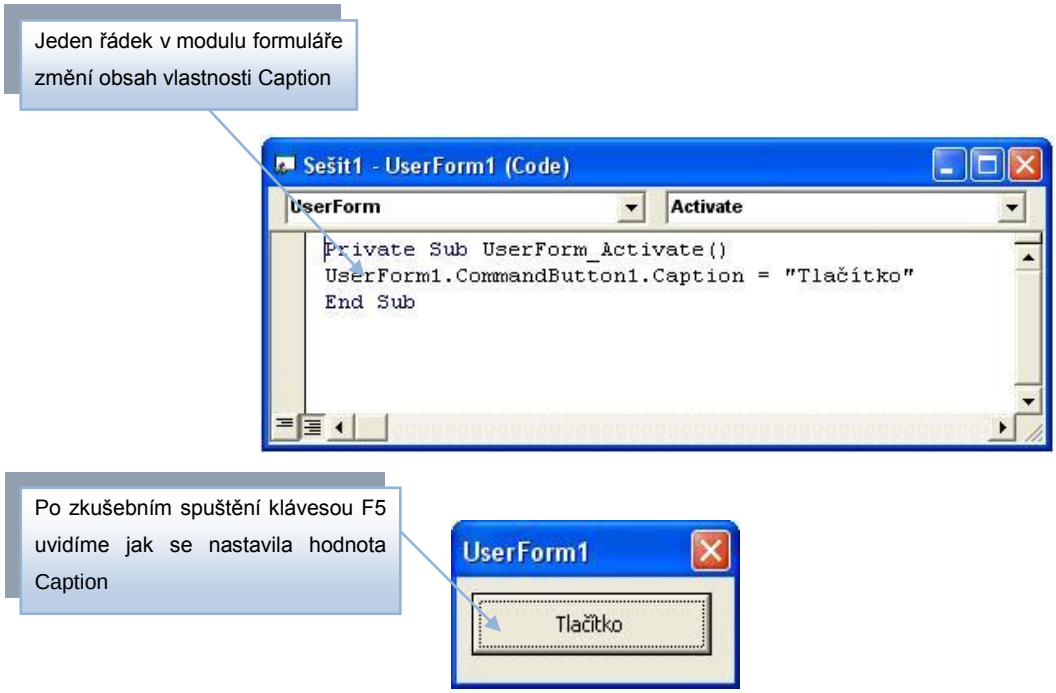


Obr. 12 Vytvoření nového objektu

Vlastnosti objektů

Vlastnosti objektů se dají rozdělit do několika skupin:

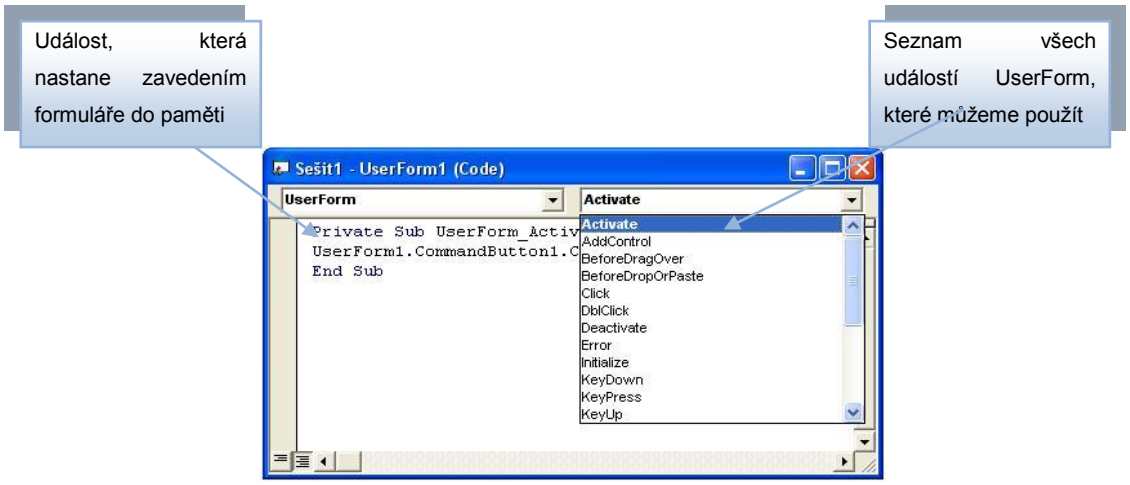
- vzhledové - určují barvu a styl zobrazení objektu
- způsobu chování - zajistí například znepřístupnění objektu na formuláři
- poziční - určují umístění a rozměry objektu
- různé - mezi ty se řadí pojmenování objektu nebo nápověda k objektu



Obr. 13 Nastavování vlastností objektů

Události objektů

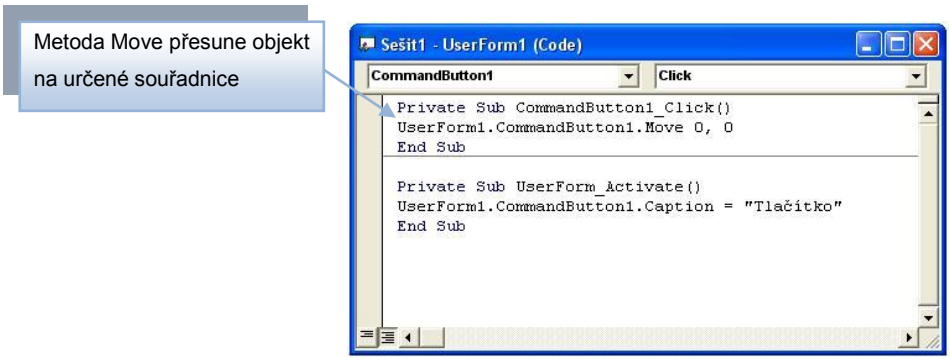
Objekty za běhu programu rozpoznávají určité události. Událostí je například pohyb kurzoru myši nad objektem nebo stisk libovolné klávesy. Každá událost může mít přiřazen námi napsaný proces, který se vykoná vždy, když daná událost nastane.



Obr. 14 Události objektů

Metody objektů

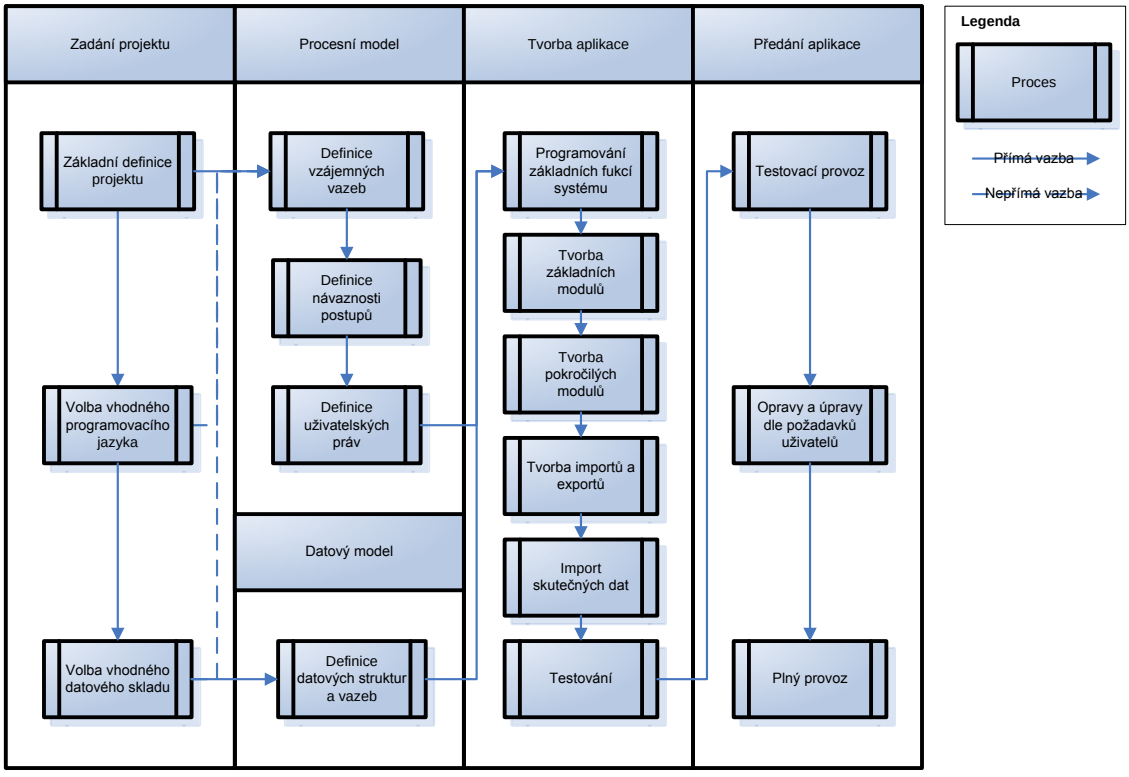
Metody jsou podobné příkazům a jsou vždy součástí nějakého objektu.



Obr. 15 Metody objektů

3 Experimentální část

Před samotnou tvorbou programu bylo potřeba provést datový a procesní návrh systému. Tento návrh spočíval v definici dat, vazeb a procesů, které by měl navrhovaný systém obsahovat. Bylo třeba přesně definovat prostup informací systémem a jejich logické vazby závislé na typu, charakteru a jejich časové souslednosti. Dále bylo třeba definovat vlastní strukturu programu s ohledem na jeho funkční nezávislost. Systém byl už od počátku koncipován jako systém modulární. Výhodou takového přístupu je jeho snadné rozšiřování a přehlednost, a to jak aplikační, tak i datová. Systém obsahuje 9 základních modulů, a to: Klienti, Investoři, Projekty, Objekty, Jednotky, Typy jednotek, SOSB/KS, Platby, Reklamace. Další částí jsou reporty, tvorba smluv do formátu Microsoft Word a modul pro nastavení práv. Všechny tyto moduly jsou mezi sebou logicky propojené.

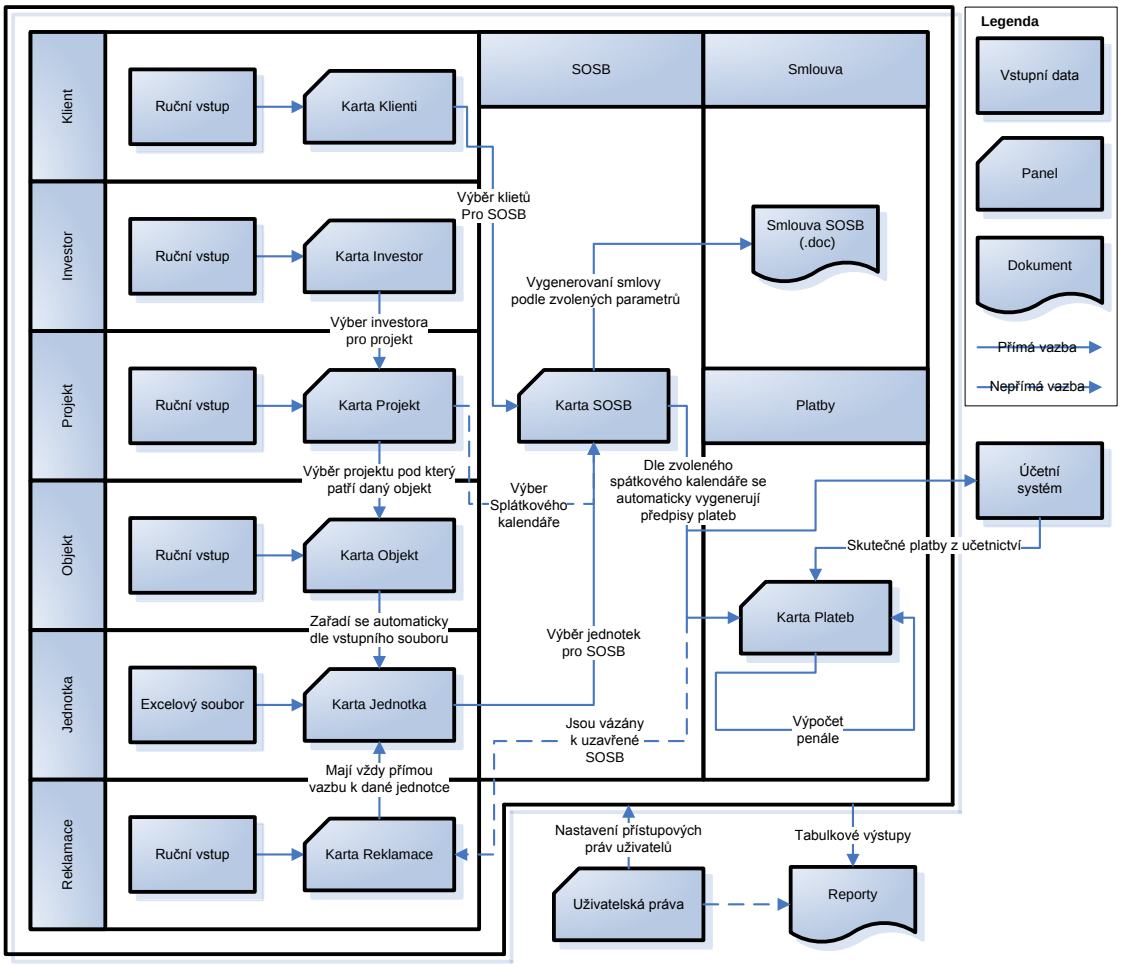


Obr. 16 Postup tvorby aplikace

3.1 Procesní model

Úkolem procesního modelu je definice a návrh funkčních vztahů jednotlivých částí programu. Samotný návrh procesního modelu vzniká již před implementací programu. K návrhu takovéhoto modelu je nutno pochopit celý proces, jenž by měl program umožňovat, včetně všech jeho vazeb k ostatním systémům (ostatními systémy jsou v tomto případě myšleny například zdroje dat pro aplikaci a stejně tak i výstupy z aplikace pro jiné systémy). Další částí, kterou by měl procesní model obsahovat, jsou časové souslednosti kroků tvorby informací v aplikaci, nutné např. ke korektní práci s programem. Poslední částí procesního modelu je definice úrovní uživatelských práv.

Nejdůležitější a zároveň i nejsložitější částí celého modelu je definice tvorby SOSB. Výsledkem je vygenerování předpisů plateb dle splátkového kalendáře a vygenerování SOSB. Před samotným vygenerováním takovéto smlouvy je třeba naplnit databázi potřebnými údaji o klientech, daném projektu, atd. Při samotné tvorbě SOSB si obsluha programu vybere prodávanou jednotku/ky, klienta/ty, splátkový kalendář daný typem projektu (nebo vytvoří individuální). Finálním krokem je pak vytvoření samotné smlouvy ve formátu MS Word a vygenerování předpisů plateb dle splátkového kalendáře. Po naimportování dat z účetního programu, systém dále umožňuje výpočet penále dané nedodržením předepsaných plateb.



Obr. 17 Procesní model

3.2 Datový model

Jako datové úložiště byl zvolen databázový systém MS Access, a to z několika důvodů. Mezi hlavní důvody patří jeho bezproblémový chod, dostatečná kapacita a nulová pořizovací cena. Na rozdíl od jiných databázových systémů (např.: MS SQL, MySQL, Oracle, atd.) ho není nutné instalovat. MS Access totiž neběží jako standardní služba operačního systému MS Windows, ale funguje jako souborová databáze. To znamená, že program s databází nekomunikuje pomocí protokolu TCP/IP a ODBC, ale přímo. Tím odpadá nastavování těchto služeb jak na straně serveru, tak i na straně klienta. Stačí pouze v programu nastavit, s jakou databází má pracovat. Tím je umožněno pracovat jak s lokální databází, tak i databází uloženou na serveru.

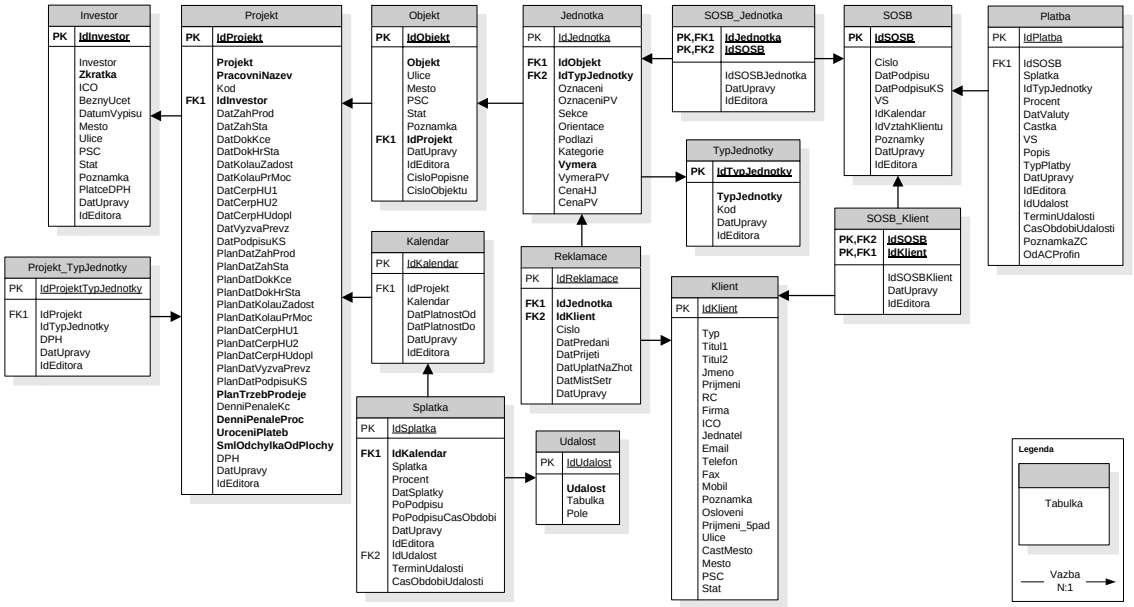
Další výhodou MS Access je možnost rozdělovat databázi na serverovou a klientskou část. V praxi to znamená, že hlavní databázový soubor je na serveru a jeho kopie je na jednotlivých klientech. Tyto kopie jsou propojeny s hlavní databází, se kterou si vyměňují pouze informace o změnách. Tak je práce uživatele se systémem mnohem rychlejší, než kdyby přistupoval k databázi na serveru, protože odpadá čas nutný ke komunikaci přes síť. I když jsou v dnešní době síťové připojení velmi rychlé, nedosahují rychlosti lokálních pevných disků. Navíc takto lze k databázi přistupovat i vzdáleně (například pomocí VPN) z jiné než lokální sítě.

MS Access funguje stejně jako jakýkoli jiný relačně databázový systém. Tzn. data jsou uložena v tabulkách. Tabulky mají strukturu podobnou, jako známe například z tabulek MS Excel. Sloupce obsahují jednotlivé ukazatele a řádky jednotlivé záznamy. Pro každý záznam (řádek) je přidán tzv. „primární klíč". Primární klíč je pro každý záznam unikátní, proto se používá jako jednoznační identifikátor dané položky a využívá se k definici vzájemných vazeb mezi tabulkami takzvanými relacemi. Relace pracují na základě porovnání dat v klíčových polích (obvykle v polích se stejným názvem v obou tabulkách). Ve většině případů se jedná o pole primárního klíče jedné tabulky, které poskytuje jedinečný identifikátor každého záznamu, a pole cizího klíče (cizí klíč vyjadřuje, jak spolu tabulky souvisejí) druhé tabulky. Existuje několik typů relací, a to: relace 1:N, relace N:N a relace 1:1.

MS Access není pouze samotná relační databáze, ale umožňuje i vytváření programu, který s těmito daty pracuje. Jako v ostatních produktech skupiny MS Office je i MS Access obsažen VBA, jež umožňuje tvorbu kódu, kterým se bude program řídit. Dále obsahuje program nástroje pro tvorbu sql dotazů, formulářů, sestav atd.

Na obr. 18 je schéma databáze. Bylo by zbytečné zde popisovat přesně jednotlivé tabulky a jejich vazby. Pokud toto schéma

porovnáme s procesním modelem, uvidíme zcela zřejmou podobnost vyplývající z logiky konstrukce programu.



Obr. 18 Datový model

(z některých tabulek byly do obrázku odebrány řádky)

3.3 Programový model

Program se skládá ze tří základních částí:

1. Hlavní částí programu je takzvané jádro, které obsahuje zdrojový kód ve formě maker a stará se o komunikaci s databází. Jsou v něm uloženy všechny formuláře používané programem a především zabezpečuje samotné řízení celé aplikace. Jádro programu je doplňkem aplikace Microsoft Excel ve formátu „xla". Výhodou doplňku je možnost pouštět takovýto soubor více uživateli.

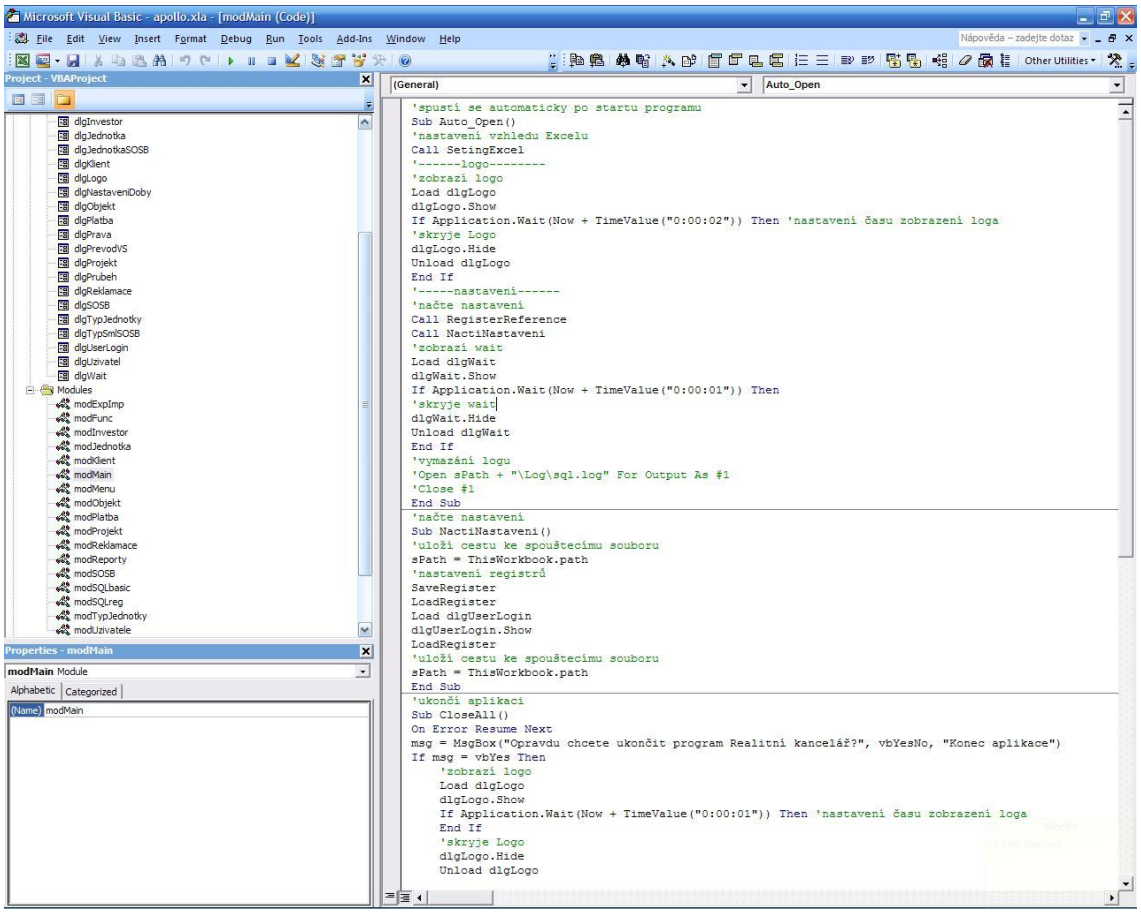
   Statisticky jádro obsahuje:

   22 formulářů

   17 modulů

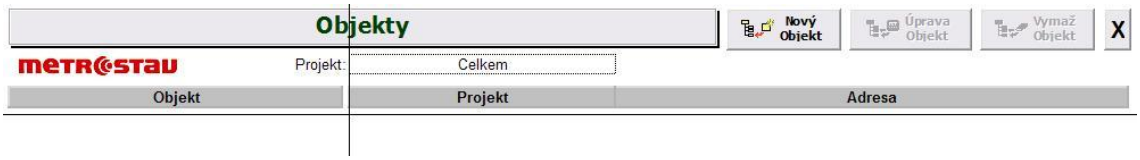
   352 procedur

   9909 řádků zdrojového kódu.



Obr. 19 Jádro aplikace zobrazené ve VBA

2. Další součástí aplikace jsou excelové listy obsahující přednastavené formáty pro zobrazení jednotlivých typů informací, takzvané panely. Panely samy o sobě žádná data neobsahují, slouží pouze k jejich zobrazování, podobně jako formuláře v aplikaci. O naplnění dat do panelu se stará jádro, které dle použitého panelu a nastavení načte aktuální data z databáze. Panely jsou šablonou aplikace Microsoft Excel ve formátu „xlt". Výhodou šablon je možnost pouštět takovýto soubor více uživateli, při každém spuštění šablony se vytvoří nový excelový soubor ve formátu „xls", kde je k původnímu názvu souboru přidán číselný index. Dalším důvodem použití formátu „xlt" je omezení možnosti uživatele nějakým způsobem tento soubor nevhodně změnit.



Obr. 20 Příklad panelu

3. Poslední, ale stejně důležitou částí je datový soubor MS Access „Apollo.mdb", který obsahuje veškerá data pro aplikaci. Kromě samotných údajů obsahuje i nastavení práv k aplikaci pro jednotlivé uživatele a další informace pro nastavení programu.

3.4 Příklady použitého kódu

Protože program obsahuje téměř 10 000 řádků zdrojového kódu, bude tato kapitola věnována pouze jeho vybraným částem. Takovým, které jsou nějakým způsobem zajímavé a to buď z pohledu nestandardního přístupu k řešení problému, nebo ukazující zajímavé možnosti využití programovacího prostředí Visual Basic for Application jako kvalitního nástroje pro vytváření aplikací pro Microsoft Office.

3.4.1 Připojení k databázi Microsoft Access

Program Microsoft Excel umožňuje přímo v programu nastavit připojení k databázi Microsoft Access a načíst z ní aktuální data. Toto připojení se provádí uživatelsky přes menu programu. K našemu účelu ale tento způsob využít nelze a to hned z několika důvodů:

- Není možné takovéto připojení ovládat pomocí zdrojového kódu
- Připojení je vázáno pouze k počítači, na kterém bylo nastaveno
- Při změně umístění databáze není možné jednoduše uživatelsky tuto cestu změnit

Abychom tyto problémy vyřešili bylo nutné vytvořit připojení k databázi pomocí zdrojového kódu. Jako nástroj pro komunikaci s databází bylo použito standardního rozhraní ADODB, pomocí

systémové knihovny Microsoft ActiveX Data Objects Recordset 2.8 Library (msador15.dll). Samotné použití této knihovny umožňuje posílat dotazy a zpracovávat data z databází podporující toto rozhraní.

Nejprve je třeba deklarovat použití objektů pro práci s databází (jsou deklarovány jako veřejné proměnné pro použití kdekoliv v aplikaci):

Public ADOConn As ADODB.Connection

Public RS As ADODB.Recordset

Public sql As String

Je třeba dále deklarovat proměnné použité přímo v proceduře

Sub SQLExecuteWithResults(sql As String)

Dim Cnct As String

Dim Connection As ADODB.Connection

Dim Recordset As ADODB.Recordset

Vytvoření konexe (proměnné Provider a DataSource jsou proměné uložené a načítané z registrů)

Set Connection = New ADODB.Connection

Cnct = "Provider=" & Provider & ";"

Cnct = Cnct & "Data Source=" & DataSource & ";"

Connection.Open ConnectionString:=Cnct

Zpracování dotazu a vytvoření objektu s načtenými daty

Set RS = New ADODB.Recordset

RS.Open Source:=sql, ActiveConnection:=Connection

End Sub

Příklad použití

sql = "SELECT IdProjekt,Projekt FROM Projekt"

Call SQLExecuteWithResults(sql)

If Not RS.EOF Then

RS.MoveFirst

While Not RS.EOF

dlgObjekt.ComboBoxProjekt.AddItem RS.Fields("Projekt")

```
    RS.MoveNext
    Wend
End If
```

3.4.2 Použití nestandardních knihoven

Při tvorbě aplikace bylo dále třeba vyřešit problém s používáním knihoven, které nejsou standardní součástí operačního systému MS Windows ani kancelářského balíku MS Office. V aplikaci je těchto knihoven celkem 6. Knihovny je před použitím potřeba zaregistrovat do systému. V operačním systému MS Windows je k tomu určen program Dll Register Server. Syntaxe jeho použití je následující:

regsvr32[/u][/s][/n][/i[:příkazový_řádek]] název_knihovny_DLL

kde:

/u	neregistrovaný server
/s	tichý režim (nebudou zobrazena okna se zprávou)
/I	předá volitelný parametr
/n	nevolá funkci DllRegisterServer

Výsledným kódem je pro potřeby aplikace:

RegisterRef = Shell("regsvr32.exe /s " & ThisWorkbook.path & "\Lib\msador15.dll", vbHide)

kde:

Shell	příkaz VBA spustí Command line
Regsvr32.exe /s	volání programu Dll Register Server v tichém režimu
ThisWorkbook.path	příkaz VBA zjistí cestu, odkud je program spouštěn

"\Lib\msador15.dll"	příkaz VBA přidá k předchozímu příkazu zbytek cesty ke knihovně
vbHide	příkaz VBA nezobrazí okno pro promt Command line

Tímto příkazem byla vyřešena samotná registrace knihovny. Pokud by takováto knihovna byla v programu přímo nastavena, kompilátor by hlásil chybu ještě před samotným spuštěním maker, čímž by nedošlo ani k jejich zaregistrování do systému. Proto je třeba knihovny do aplikace přidat dynamicky při spuštění programu. K takovému účelu má VBA příkaz object. AddFromFile(filename), výsledný kód pak vypadá takto:

ActiveWorkbook.VBProject.References.AddFromFile
ThisWorkbook.path & "\Lib\msador15.dll").

Tím se knihovny zaregistrují až po spuštění programu, aniž by došlo k zablokování aplikace.

3.5 Popis uživatelského rozhraní programu

V této části práce se budeme věnovat popisu aplikace z pohledu uživatelského rozhraní. Nebylo by účelné popisovat všechny části programu, se kterými uživatel přijde do styku, proto se zde budeme věnovat pouze vybraným panelům. Především takovým, které jsou něčím zajímavé, nebo odlišné od ostatních panelů.

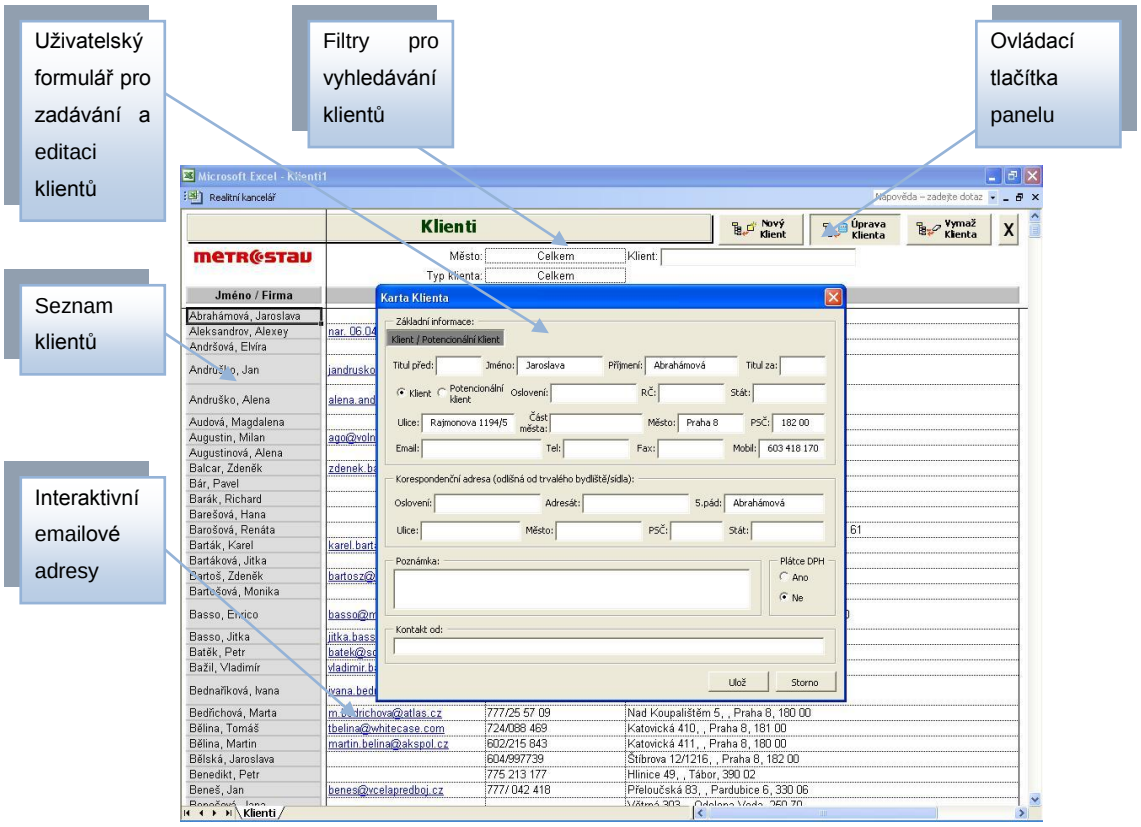
3.5.1 Panel Menu

Prvním z těchto panelů je menu aplikace. Tento panel je hlavním ovládacím prvkem programu pro uživatele. Je zobrazen jako první po přihlášení do aplikace. Z Menu je možno spouštět všechny další panely, vytvářet reporty, importovat a exportovat data a nastavovat vlastnosti programu. Menu je možné ovládat dvěma způsoby, a to buď přes standardní menu, nebo pomocí tlačítkového menu umístěného přímo v panelu. Tlačítkové menu obsahuje pouze vybrané (nejčastěji používané) ovládací prvky.

Obr. 21 Menu aplikace

3.5.2 Panel Klienti

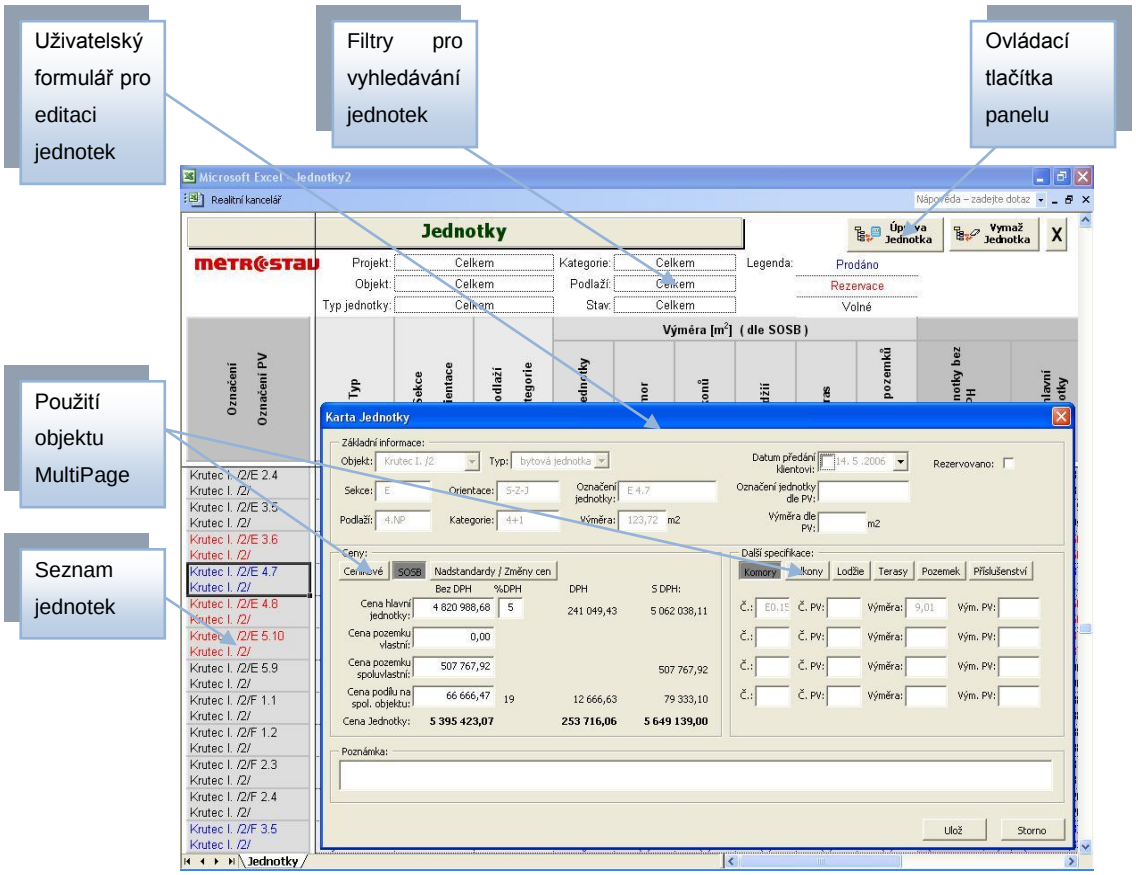
Panel Klienti obsahuje informace o všech klientech a firmách. Umožňuje tyto informace zadávat, editovat a vyhledávat. Zvláštností tohoto panelu je možnost posílat e-mailové zprávy při kliknutí na e-mailovou adresu klienta (je nutno mít nakonfigurován MS Outlook).



Obr. 22 Panel Klienti

3.5.3 Panel Jednotky

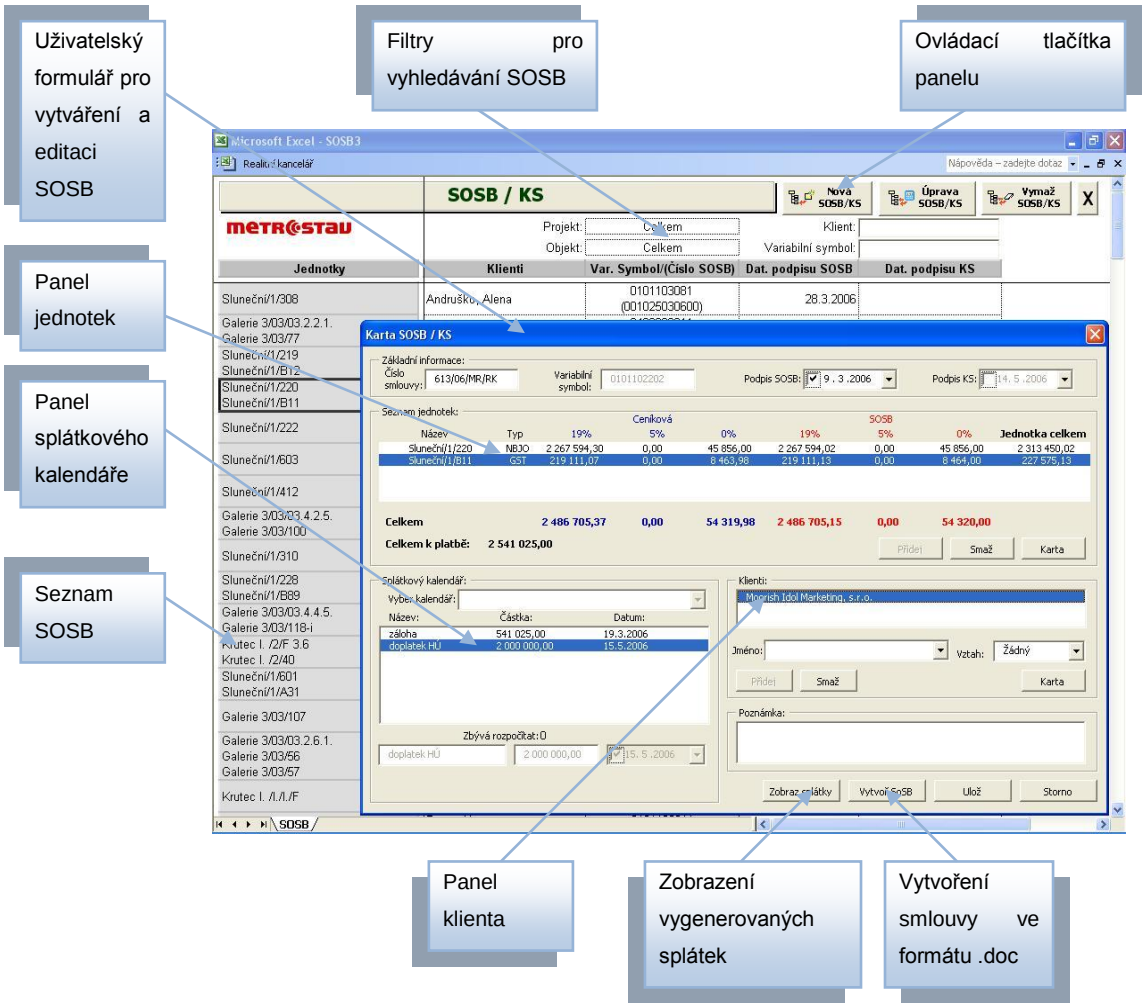
Panel jednotky slouží k editaci a vyhledávání jednotek. Zadávání nových jednotek se provádí importem z externího excelového souboru. Každá jednotka může obsahovat až 80 údajů. Aby bylo možno pracovat s tak velkým počtem údajů na jednom formuláři, je zde využit objekt MultiPage, který umožňuje pomocí tlačítek (nebo záložek) měnit zobrazení položek na formuláři.



Obr. 23 Panel Jednotky

3.5.4 Panel SOSB

Panel SOSB je nejdůležitějším panelem aplikace, umožňuje uživateli pomocí několika „kliknutí" vytvořit novou smlouvu SOSB, nebo již vytvořenou smlouvu editovat. Z formuláře tohoto panelu lze vstupovat i do karty vybrané jednotky. Dále umožňuje vytvářet smlouvu SOSB ve formátu *.doc. V neposlední řadě vytváří splátky pro danou smlouvu. Více bylo popsáno v procesním a programovém modelu.



Obr. 24 Panel SOSB

Program obsahuje celkem 17 panelů, každý z nich je přizpůsoben zobrazování informací daného typu, ale logika a formát je velmi podobný zde prezentovaným panelům.

4 Závěr

Cílem práce byl návrh a implementace systému pro zpracování dat v prostředí realitní kanceláře. Pro vytvoření aplikace takovéhoto rozsahu bylo nutno vytvořit především kvalitní návrh. Návrh systému trval cca měsíc, v tomto období byl definován jeho rozsah, potřebná funkcionalita, uživatelské prostředí, atd. Samotná implementace systému od zadání až po jeho odevzdání trvala přibližně 4 měsíce intenzivní práce. Nejprve bylo vytvořeno jádro aplikace, které obsahuje všechny základní funkce systému a stará se o komunikaci s databází. Poté byly postupně vytvářeny jednotlivé moduly aplikace dané logikou postupného zpracování informací. Jádro systému obsahuje téměř 10 000 řádků zdrojového kódu rozdělených do 17 modulů, 22 formulářů a 352 procedur a funkcí. Před samotným spuštěním programu do ostrého provozu byl podroben pečlivému testování ze strany zadavatele. V tomto období byly odstraněny drobné chyby a byl upraven dle požadavků uživatelů. V současné době je aplikace používaná v běžném provozu několik měsíců. S programem aktivně pracuje cca 15 uživatelů a je průběžně rozšiřován o nové funkce a další informace. V plánu zadavatele je i další velké rozšíření systému o tvorbu kupních smluv, takovéto rozšíření by znamenalo přibližně třetinový nárůst celkového rozsahu programu.

5 Seznam použité literaturatury

[1] Walkenbach, John: Microsoft Excel 2000 a 2002 programování ve VBA. Computer Press, Brno, 2004, ISBN 80-7226-547-4

[2] Viescas, John L.: Mistrovství v Microsoft Office Access 2003, Computer Press, Brno, 2005, ISBN 80-251-0537-7

[3] Lacko, Luboslav: SQL Hotová řešení, Computer Press, Brno, 2003, ISBN 80-7226-975-5

[4] Farana, Radim: Databázové systémy, Technická univerzita Ostrava, Ostrava, 1995, Dostupné z < http://www.fs.vsb.cz/books/dbacc20/Welcome.htm>

[5] Hladiš, Petr: Visual Basic pro začátečníky, 2001, Dostupné z <http://www.vbasic.cz/serial.asp?id=0>